AES-Based Grinding for MPC-in-the-Head Signatures

Matthieu Rivain

CryptoExperts, Paris, France
matthieu.rivain@cryptoexperts.com

Abstract. Grinding is a technique which introduces a proof of work into the Fiat-Shamir transform: by constraining the challenge to satisfy a w-bit condition, forging a proof requires about $2^w/\varepsilon$ evaluations of the hash function instead of $1/\varepsilon$, where ε is the soundness error of the underlying protocol. This allows one to select reduced parameters, yielding shorter proofs and signatures. Grinding is used in FAEST, MQOM and SDitH, the three MPC-in-the-Head schemes selected for the third round of the NIST additional post-quantum signature standardization process, where it is instantiated with Keccak. In this short paper, we investigate grinding schemes in which the proof of work is expressed in terms of block cipher computations, specifically AES, which is significantly faster than Keccak on modern CPUs, is already a building block of these schemes, and underlies the very definition of the NIST security categories. We formalize the notion of grinding scheme together with a protocol-agnostic security notion, we propose a construction performing two cipher calls per iteration, and we prove, in the ideal cipher and random oracle models, that an adversary making Q_E cipher queries breaks it with probability at most $\frac{4}{3} \cdot \varepsilon Q_E/2^w$, up to negligible terms. We further generalize the scheme to use more cipher calls per iteration, which makes the constant $\frac{4}{3}$ tend to 1.

1 Introduction

Grinding is a technique which consists in introducing a proof of work into the Fiat-Shamir transform. Recall that a cheating prover attacking a non-interactive proof obtained by Fiat-Shamir from a protocol of soundness error ε can simply hash about $1/\varepsilon$ candidate first messages, until the derived challenge falls into the set of bad challenges for which they can answer. Grinding constrains the Fiat-Shamir challenge to further satisfy a w-bit condition: obtaining a single admissible challenge now costs 2^w evaluations on average, so that forging a proof costs about $2^w/\varepsilon$ evaluations. A security level of λ bits is therefore reached with $\varepsilon = 2^{-(\lambda-w)}$ instead of $\varepsilon = 2^{-\lambda}$, which allows one to select reduced parameters and yields shorter proofs and/or a faster scheme. The honest prover, however, pays the very same 2^w evaluations: grinding trades proving time for proof size, and the practical interest of the technique therefore hinges on the *unit cost* of the underlying grinding computation.

This technique is commonly used in hash-based SNARKs (a.k.a. STARKs) [Sta21], where it reduces the number of query repetitions in the low-degree test, and it was first suggested in the MPC-in-the-Head (MPCitH) setting in [BBM+24]. In the MPCitH setting, one typically has a soundness error of the form $(d/N)^\tau$, where d is the degree of the verified constraints, N is the size of the GGM trees and τ is the number of trees (*i.e.* the number of repetitions).[1] Introducing grinding allows one to set the parameters such that $\tau \cdot \log_2(N/d) \geq \lambda - w$ instead of $\tau \cdot \log_2(N/d) \geq \lambda$, thereby saving one or two repetitions and lowering the proof (or signature) size accordingly. The cost is a short nonce appended to the proof, and a few extra operations for the verifier, both negligible with respect to the saved repetitions.

Grinding is notably employed in FAEST v2 [BBB+24], MQOM v2 [BBFR24] and SDitH v2 [ABB+24], which are the three MPCitH schemes among the nine candidates selected by NIST in May 2026 for the third round of the ongoing additional post-quantum signature standardization process. In these three schemes, grinding is performed with the Fiat-Shamir hash function itself, namely Keccak, by appending a counter to the hashed string and incrementing it until the w-bit condition is met. The load of Keccak

[1] To be precise, a soundness error of $(d/N)^\tau$ corresponds to Threshold-Computation-in-the-Head (TCitH) proofs based on parallel repetitions [FR25], whereas VOLE-in-the-Head (VOLEitH) proofs [BBD+23] achieve the smaller soundness error d/N^τ by aggregating the τ repetitions into a single one over an extension field. This distinction does not affect our discussion: in both cases the soundness level $\log_2(1/\varepsilon)$ grows linearly with τ, so that relaxing the requirement from λ to $\lambda - w$ bits saves one or two repetitions, each repetition contributing $\log_2(N/d)$ bits in the former case and $\log_2 N$ bits in the latter.

computations in the Fiat-Shamir transform is thus increased by a factor 2^w, which is what limits the grinding parameter w in practice.

In this short paper, we investigate grinding schemes in which the proof of work is expressed in terms of *block cipher* computations – specifically AES – rather than hash computations, the adversary being constrained to perform about $2^w/\varepsilon = 2^\lambda$ cipher calls. This is motivated by three observations. First, AES is significantly faster than Keccak on modern CPUs thanks to dedicated instruction sets, and the grinding loop is embarrassingly parallel, so that the high throughput of pipelined AES instructions can be fully exploited. Preliminary benchmarks suggest that one AES-128 computation is roughly 10 times faster than one Keccak-f[1600] permutation on a modern CPU, when amortized over several computations, which translates into a gain of about 3 bits on the grinding parameter w for the same signing overhead. Second, AES is already a building block of the above schemes, where it is used for the GGM seed trees, the seed expansion and the seed commitments: no new primitive and no new assumption is introduced, and optimized constant-time implementations are already available in these schemes. Third, the NIST post-quantum security categories are themselves defined in terms of the cost of exhaustive key search on AES, so that measuring the hardness of grinding as a number of AES computations is the natural metric for these schemes.

Replacing the hash function by a block cipher is not a mere substitution. A block cipher has a small and rigid interface: a single call offers only 2λ bits of input, which must simultaneously accommodate a digest of the hashed string x, the grinding nonce, and the bits ensuring domain separation from the other cipher calls of the signature scheme. It is also invertible, so that the adversary may additionally make backward queries. Finally, since the digest of x can only be partially bound into a call, an adversary may amortize a single cipher call over several strings x, which costs a small constant factor in the security bound. Our construction addresses these issues by deriving two (or $t > 2$) cipher calls per attempt from the hashed string, by defining the success condition jointly on the two (or $t > 2$) ciphertexts, and by binding the ciphertexts into the resulting Fiat-Shamir challenge.

We first formalize the notion of a *grinding scheme* and the associated security notion (Section 2). This notion is agnostic to the underlying protocol so that it can be plugged into existing proofs of signature schemes or non-interactive arguments. We then propose a grinding scheme based on a block cipher (Section 3), for which the honest prover performs 2^w cipher calls on average, and we prove (Section 4) that in the ideal cipher and random oracle models any adversary making Q_E cipher queries wins with probability at most $\frac{4}{3} \cdot \varepsilon Q_E/2^w$, up to negligible terms. This is tight as a matching attack exists which exploits collisions in the values derived from the hashed strings. We finally describe two variants of the scheme: one using $t = 2^\tau$ cipher calls per attempt, which keeps the honest prover's cost of 2^w calls while replacing the constant $\frac{4}{3}$ by $\frac{2t}{2t-1}$, tending to 1 as t grows; and one adapted to the second configuration of the cipher calls in the original scheme, in which the secret material lies in the key argument.

2 Grinding

2.1 Grinding Scheme

Let us start by formalizing the notion of grinding scheme. A *grinding scheme* with security level $\lambda \in \mathbb{N}$, grinding parameter $w \in \mathbb{N}$ and nonce length $\ell \in \mathbb{N}$, is a pair of algorithms $(\mathsf{Grind}, \mathsf{Check})$:

- $\mathsf{Grind}(x) \to (\rho, h)$: an algorithm that, on input a string $x \in \{0,1\}^*$, outputs a nonce $\rho \in \{0,1\}^\ell$ and a digest $h \in \{0,1\}^{2\lambda}$, or a failure symbol $\perp$ if no valid nonce can be found.
- $\mathsf{Check}(x, \rho) \to h$ or $\perp$: a deterministic algorithm that recomputes the digest h if ρ is a valid nonce for x, and outputs $\perp$ otherwise.

Correctness requires $\mathsf{Check}(x, \rho) = h$ for all $(\rho, h) \leftarrow \mathsf{Grind}(x)$ and $\Pr[\perp \leftarrow \mathsf{Grind}(x)] \leq 2^{-\lambda}$ (formally, since Grind is deterministic, the latter might only hold in an ideal model over the randomness of the oracles). Informally, the security requires that no adversary making Q queries to the underlying hash and/or cipher oracles outputs a valid nonce for a fresh x with probability greater than $Q/2^w$ (up to negligible terms). This security requirement is formalized in Section 2.2.

The idea of grinding is to replace the computation of a Fiat-Shamir digest $h = H(x)$ in the signing (or proving) algorithm by a call $(\rho, h) \leftarrow \mathsf{Grind}(x)$, where h is used as the challenge in place of $H(x)$ and the nonce ρ is appended to the signature (or proof). The verification algorithm correspondingly replaces the recomputation of $H(x)$ by $h = \mathsf{Check}(x, \rho)$, rejecting whenever Check returns $\perp$.

2.2 Grinding Security Model

Grinding is used in the context of signature schemes and/or arguments of knowledge to increase the security while applying the Fiat-Shamir transform. To motivate the security model, we give a brief overview of the Fiat-Shamir transform and its security implications.

A simple Fiat-Shamir security game. Consider a Σ-protocol $(\mathcal{P}, \mathcal{V})$ for a relation R where $\mathcal{P}$ is the prover and $\mathcal{V}$ is the verifier. The protocol consists of three moves: the prover sends a commitment c, the verifier sends a challenge h, and the prover responds with a response r. The verifier then checks whether (c, h, r) is valid with respect to a statement x for the relation R. We consider here a public-coin Σ-protocol, where the challenge h is a bit string chosen uniformly at random by the verifier from the space $\{0,1\}^{2\lambda}$, which can then be used as a random tape to sample a structured challenge. When applying the Fiat-Shamir transform, the challenge h is replaced by a hash of a string x that includes the commitment c, the message (in a signature scheme) and possibly other public information.

The soundness of the Σ-protocol can generally be expressed as follows: for any given commitment c, there exists a space of bad challenges $\mathcal{H}_c \subseteq \{0,1\}^{2\lambda}$ such that for any challenge $h \notin \mathcal{H}_c$, the prover cannot produce a response r such that (c, h, r) is valid for the statement x, unless they know some witness w such that $(\mathsf{x}, \mathsf{w}) \in R$ (this knowledge property is captured by exhibiting an extractor which upon one or several valid transcripts (c, h, r), with $h \notin \mathcal{H}_c$, can extract a witness). Denoting

$$\varepsilon := \max_c \frac{|\mathcal{H}_c|}{2^{2\lambda}}$$

the soundness error of the Σ-protocol, we have that for any commitment c, the probability that a random challenge h is in $\mathcal{H}_c$ is at most ε. We can then define a simple security game for the Fiat-Shamir transform as follows: an adversary $\mathcal{A}$ is given a hash function $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$ and outputs a string x (of prescribed format). The adversary wins if the challenge $h = H(x)$ is in $\mathcal{H}_c$ for some commitment c included in x. When H is modeled as a random oracle, it is not hard to see that the adversary's advantage in this game is at most $Q_H \cdot \varepsilon$, where Q_H is the number of queries made to the random oracle H.

Most of the security proofs for signature schemes (or NIZKs) using Fiat-Shamir in the random oracle model rely on this simple security game as part of the proof (which further needs to be combined with the knowledge extractor for the concrete protocol). Our security model for grinding is a generalization of this simple security game, where the adversary wins if it outputs a string x and a nonce ρ such that $\mathsf{Check}(x, \rho) = h \neq \perp$, with h in the space of bad challenges $\mathcal{H}_c$ for some commitment c included in x.

Grinding security game. We formally define the grinding security in the following definition. As this notion is agnostic to any concrete protocol, underlying relation R, and commitment format, we simply define the grinding security for any set of bad challenges $\mathcal{H}_x$ for every string x such that $|\mathcal{H}_x| \leq \varepsilon \cdot 2^{2\lambda}$ for some soundness parameter $\varepsilon \in [0,1]$. Our security model is then defined in the ideal cipher model and random oracle model.

Definition 1 (Grinding security). *Let $\lambda, w, \ell \in \mathbb{N}$ be the security parameter, the grinding parameter and the nonce length, respectively. Let $E : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a block cipher modeled as an ideal cipher and $H : \{0,1\}^* \to \{0,1\}^*$ be an (extendable output) hash function modeled as a random oracle. For every bit string $x \in \{0,1\}^*$, let $\mathcal{H}_x \subseteq \{0,1\}^{2\lambda}$ some set such that $|\mathcal{H}_x| \leq \varepsilon \cdot 2^{2\lambda}$ for some $\varepsilon \in [0,1]$. We say that the grinding scheme $(\mathsf{Grind}, \mathsf{Check})$ is (Q_E, Q_H, ε')-grinding secure if for any adversary $\mathcal{A}$ making at most Q_E queries to the block cipher oracle E, at most Q_H queries to the hash oracle H, the probability that $\mathcal{A}$ outputs a string x^* and a nonce ρ^* such that $\mathsf{Check}(x^*, \rho^*) = h^* \neq \perp$ with $h^* \in \mathcal{H}_{x^*}$, is at most ε'.*

Intuitively, the target security for a grinding scheme is to achieve $\varepsilon' \approx Q \cdot \varepsilon / 2^w$ for $Q = Q_E + Q_H$, possibly up to a small constant factor and negligible terms.

3 Proposed Grinding Scheme

We propose a grinding scheme based on a block cipher (typically AES). We assume that the scheme is to be used in a context where the block cipher might be involved as a subroutine for other operations (typically for seed tree, seed commitment and seed expansion in an MPC-in-the-Head signature scheme). In such a context, two configurations are possible:

1. The block cipher calls are made with a key $k = \mathsf{salt} \parallel \mathsf{tweak}$ and a (secret) plaintext p;
2. The block cipher calls are made with a plaintext $p = \mathsf{salt} \parallel \mathsf{tweak}$ and a (secret) key k;

where salt is a public salt, unique to a given signature, and tweak is a tweak ensuring domain separation between the different block cipher calls. In the following, we consider the first configuration. The second configuration is addressed in Section 5.

Applying our grinding scheme requires adding a selector bit to the tweak, which is set to 0 for the block cipher calls in the signature scheme and to 1 for the grinding scheme. This ensures domain separation between the grinding scheme and the other block cipher calls. Since the tweak space is defined by the signature scheme, this implies dropping one bit from the salt i.e. halving the salt space. Security-wise, this slightly increases the salt collision probability, which is not a problem in practice.

3.1 Description

The scheme is parametrized by the security parameter λ, the grinding parameter w, and the nonce length ℓ. It relies on a block cipher

$$E : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda,$$

and a hash function $H : \{0,1\}^* \to \{0,1\}^*$, from which we define the domain-separated functions

$$H_0 : \{0,1\}^* \to \{0,1\}^{2(\kappa+\theta)}, \qquad H_1 : \{0,1\}^* \to \{0,1\}^{2\lambda}, \qquad H_i(\cdot) = H(i \,\|\, \cdot)\ ,$$

where κ and θ are the "free bits" of the key and plaintext parts of the cipher calls, respectively. In our construction, we input two selector bits in the key argument and ℓ nonce bits in the plaintext argument of the cipher calls, so

$$\kappa = \lambda - 2, \qquad \theta = \lambda - \ell.$$

We keep κ and θ symbolic throughout to allow easy generalization to other cipher sizes and/or placement of the selector bits. We write $[x]_n$ for the n least significant bits of x and $\parallel$ for concatenation.

Domain separation. Two selector bits are appended to the key. The outermost (last) bit is shared with the signature scheme: keys used by the signature scheme (e.g. for GGM trees) end with the bit 0 (one selector bit, so the signature scheme sacrifices only one bit), while the grinding scheme uses keys ending with the two bits $(j, 1)$, where the inner bit $j \in \{0,1\}$ indexes the two cipher calls.

Derived values. For a string $x \in \{0,1\}^*$, parse

$$H_0(x) = (g_0, g_1, k_0, k_1) \in \{0,1\}^\theta \times \{0,1\}^\theta \times \{0,1\}^\kappa \times \{0,1\}^\kappa,$$

and define, for a nonce $\rho \in \{0, \ldots, 2^\ell - 1\}$, the two calls

$$c_0 := E\big(k_0 \| 01,\ g_0 \| \rho\big), \qquad c_1 := E\big(k_1 \| 11,\ g_1 \| \rho\big).$$

The nonce ρ enters the *plaintext* of *both* calls; the key parts k_0, k_1 and block parts g_0, g_1 are fixed by x.

Success condition. The grinding scheme succeeds if the two ciphertexts c_0, c_1 satisfy the condition

$$[c_0 \oplus c_1]_{w-1} = 0,$$

i.e. if the $w - 1$ least significant bits of the XOR of the two ciphertexts are all zero.

Description. The grinding scheme is summarized in Figure 1. The Grind function attempts to find a nonce ρ such that the success condition is satisfied, and returns the nonce along with a hash h defined as $h := H_1(x \| \rho \| c_0 \| c_1)$ in case of success. The Check function verifies whether a given nonce ρ satisfies the success condition for a given input x.

We chose to use two cipher calls per iteration in the grinding scheme, which we found to be a sweet spot between security and efficiency/simplicity. We discuss the generalization to $t = 2^\tau$ cipher calls per iteration in Section 5.

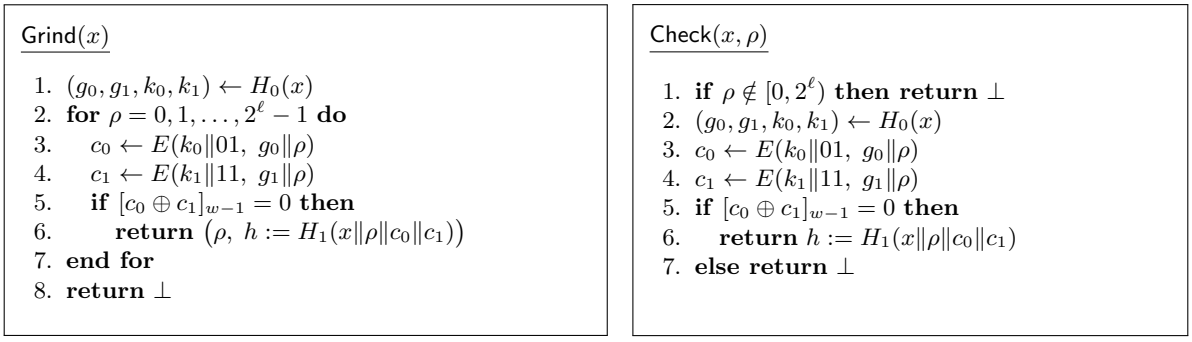

Fig. 1. Grinding scheme.

Remark 1 (Correctness). For any x, the probability that $\mathsf{Grind}(x)$ fails to find a valid nonce is upper bounded by

$$\Pr[\perp \leftarrow \mathsf{Grind}(x)] \ \le\ \left(1 - 2^{-(w-1)}\right)^{2^\ell} \ \le\ e^{-2^{\ell-w+1}},$$

at most $2^{-\lambda}$ whenever $\ell \ge w - 1 + \log_2(\lambda \ln 2)$ ($\approx w + 5.5$ for $\lambda = 128$). Note that we would have the same bound if E was a random function instead of a random permutation. Here, the without-replacement sampling induced by the random permutation can only *lower* the failure probability. We stick to this simpler upper bound for concreteness which is sufficient in practice.

Remark 2 (Full nonce in both calls). The entire nonce must enter both plaintexts. Splitting it (e.g. ρ_1 in call 0, ρ_2 in call 1) would make the two calls depend on disjoint nonce halves, turning grinding into a birthday-type search over the grid $\rho_1 \times \rho_2$ and destroying the 2^w factor.

4 Security Proof

We use the grinding security game of Definition 1, in the ideal cipher model and random oracle model, with the bad-challenge family $\{\mathcal{H}_x\}_x$ fixed independently of the oracles.

Theorem 1. *Let Q_E, Q_{H_0}, Q_{H_1} be the maximum numbers of queries to E, H_0, H_1, respectively. Let $\mu \in \mathbb{N}$ and $\alpha := Q_E/(2^\lambda - Q_E)$. The grinding scheme of Figure 1 is (Q_E, Q_H, ε')-secure with $Q_H = Q_{H_0} + Q_{H_1}$ and*

$$\varepsilon' \ \le\ \varepsilon \left(\frac{4}{3} \frac{Q_E}{2^w} + \frac{\mu 2^\ell Q_E}{2^\lambda} + \mu 2^\ell \alpha + \frac{\mu Q_E Q_{H_0}^3}{2^{2(\kappa+\theta)}} + \frac{Q_{H_0} Q_E^2}{2^{2(\kappa+\theta)+2}} + \frac{5\,(1+\alpha)\,Q_{H_1}}{2^\lambda} \right) + p_{\mu+1}$$

where $p_{\mu+1}$ is the probability of a $(\mu+1)$-fold collision among $2Q_{H_0}$ uniform κ-bit values which satisfies:

$$p_{\mu+1} \le \frac{\binom{2Q_{H_0}}{\mu+1}}{2^{\mu\kappa}} \le 2^\kappa \left(\frac{2eQ_{H_0}}{(\mu+1)2^\kappa} \right)^{\mu+1} .$$

Concrete values for α and μ. Assuming $Q_E \le 2^{\lambda-1}$ (reasonable in practice), we have $\alpha \le 1$. The parameter μ is chosen so that $p_{\mu+1} \le 2^{-\lambda}$. Assuming $Q_{H_0} \le 2^{\lambda-c}$ for some $c \ge 0$ and substituting $\kappa = \lambda - 2$ (so that $2^\kappa / Q_{H_0} \ge 2^{c-2}$ and $\lambda + \kappa = 2\lambda - 2$), it is enough to take μ satisfying

$$(\mu+1)\left(\log_2(\mu+1) + c - 3 - \log_2 e \right) \ \ge\ 2\lambda - 2 .$$

As the left-hand side is increasing in μ, any μ meeting this inequality is admissible; the smallest such μ is $O(\lambda)$ and decreases as the query margin c increases. For instance, at $\lambda = 128$ one may take $\mu = 109$ for $c = 0$ (the full regime $Q_{H_0} \le 2^\lambda$), $\mu = 85$ for $c = 1$, or $\mu = 63$ for $c = 3$.

Interpretation of the bound. With the above choices we have $p_{\mu+1} \le 2^{-\lambda}$ and $\alpha \le 1$, and the bound of Theorem 1 is dominated by its first term $\varepsilon \cdot \frac{4}{3} \cdot \frac{Q_E}{2^w}$. Indeed, taking the worst case $Q_E, Q_{H_0}, Q_{H_1} \le 2^\lambda$ (all tending to 2^λ) and recalling $2(\kappa+\theta) = 4\lambda - 2\ell - 4$, the parenthesized terms other than the first sum to

$$\le\ \mu 2^\ell(1+\alpha)\ +\ \mu 2^{2\ell+4}\ +\ \frac{2^{2\ell+2}}{2^\lambda}\ +\ 5(1+\alpha)\ .$$

The dominant residual term $\mu 2^{2\ell+4}$ is negligible compared to $\frac{4}{3}Q_E/2^w$ as soon as $2\ell + w + \log_2 \mu \ll \lambda$, which is comfortably satisfied for the relevant parameters (e.g. $\ell \approx w+6$, $w \le 20$, $\mu \le 128$). Hence the scheme achieves the target security $\varepsilon' \approx \frac{4}{3} \cdot \varepsilon Q_E/2^w$ up to negligible terms.

Proof of Theorem 1

Conventions. We denote $\mathcal{Q}_E$, $\mathcal{Q}_{H_0}$, $\mathcal{Q}_{H_1}$ the set of query-response records made by $\mathcal{A}$ to E, H_0, H_1. The total number of queries is bounded by Q_E, Q_{H_0}, Q_{H_1}, respectively. We call a query to E a *forward* cipher query and a query to E^{-1} a *backward* cipher query (both recorded in $\mathcal{Q}_E$). The *transcript* of $\mathcal{A}$ is the ordered list of all query-response records, in the order they were made. Without loss of generality:

- $\mathcal{A}$ never repeats a query. Cipher queries in either direction are recorded as *records* $(k, u, v) \in \mathcal{Q}_E$ meaning $E_k(u) = v$, together with their direction and transcript position.
- All cipher queries use keys with selector bits $(j, 1)$, $j \in \{0,1\}$: queries with last selector bit 0 concern independent permutations which do not affect the outcome of the grinding game.
- The queries computing $\mathsf{Check}(x^*, \rho^*)$ on the output — one query to H_0, 2 queries to E and one query to H_1 — are counted among the Q_E, Q_{H_0}, Q_{H_1} budgets (an adversary omitting them is turned into one that makes them, at the cost of 2 cipher and 1+1 hash queries). After this completion the win condition is fully determined by the query transcript.

Tuples, links, certification. For each record $(x, (g_0, g_1, k_0, k_1)) \in \mathcal{Q}_{H_0}$ define the two *tuples*

$$A(x) := (k_0, g_0) \in \{0,1\}^{\kappa+\theta}, \qquad B(x) := (k_1, g_1) \in \{0,1\}^{\kappa+\theta}.$$

A cipher record (k, u, v) with $k = z\|01$ and $u = y\|\rho$ (key part $z \in \{0,1\}^\kappa$, block part $y \in \{0,1\}^\theta$, nonce ρ) is *A-relevant for x at ρ* if $(z, y) = A(x)$; B-relevance is defined symmetrically with selector $(1,1)$ and $B(x)$. For fixed (x, ρ) there is at most one A-relevant and one B-relevant record (key, plaintext and hence the record are determined). Crucially, an A-relevant or B-relevant record for x at ρ already commits to ρ through its plaintext.

A pair (x, ρ) is *linked* if $\mathcal{Q}_{H_0}$ contains a record $(x, (g_0, g_1, k_0, k_1))$ and $\mathcal{Q}_E$ contains both an A-relevant record $(k_0\|01, g_0\|\rho, c_0)$ and a B-relevant record $(k_1\|11, g_1\|\rho, c_1)$ for x at ρ; it is *certified* if moreover $[c_0 \oplus c_1]_{w-1} = 0$, i.e. $\mathsf{Check}(x, \rho) \neq \perp$. Let L be the number of linked pairs and $N \le L$ the number of certified pairs. Each linked pair is witnessed by three records:

- the H_0 record $(x, (g_0, g_1, k_0, k_1))$,
- the A-relevant cipher record $(k_0\|01, g_0\|\rho, c_0)$,
- the B-relevant cipher record $(k_1\|11, g_1\|\rho, c_1)$.

Among the three records, the last appearing in the transcript is called the *completing record* of the linked pair, created by its *completing query*. Blind cipher queries (made before the matching H_0 query) are permitted and are handled by the case where the completing record is the H_0 record. The A-relevant and B-relevant records of a linked pair are further said to be *partner* records of each other for that pair.

Bad events. We define two bad events that can occur in the transcript of $\mathcal{A}$'s queries, which are used to bound the number of linked and certified pairs:

- $\mathsf{MultKey}$: some value $z \in \{0,1\}^\kappa$ appears more than μ times in the multiset of the $2Q_{H_0}$ key parts $\{\!\{k_0, k_1\ ;\ (x, (g_0, g_1, k_0, k_1)) \in \mathcal{Q}_{H_0}\}\!\}$.
- Struct: among the $2Q_{H_0}$ tuples $A(x), B(x)$,
  (i) some tuple $(z, y) \in \{0,1\}^{\kappa+\theta}$ occurs ≥ 3 times in the multiset $\{\!\{A(x), B(x)\ ;\ (x, ...) \in \mathcal{Q}_{H_0}\}\!\}$, or

(ii) some x is *doubly-collided*: $A(x)$ occurs twice in the multiset $\{\!\{A(x)\,;\,(x,...)\in\mathcal{Q}_{H_0}\}\!\}$ and $B(x)$ occurs twice in the multiset $\{\!\{B(x)\,;\,(x,...)\in\mathcal{Q}_{H_0}\}\!\}$.

Lemma 1 (Bad event probabilities). *We have:*

$$\Pr[\mathsf{MultKey}]\le p_{\mu+1},\qquad \Pr[\mathsf{Struct}]\le \frac{\frac{4}{3}Q_{H_0}^3}{2^{2(\kappa+\theta)}}\ .$$

Proof. MultKey: The $2Q_{H_0}$ key parts are uniform independent κ-bit strings; a $(\mu+1)$-fold collision has probability $p_{\mu+1}$ by definition. The bound $p_{\mu+1}\le\binom{2Q_{H_0}}{\mu+1}2^{-\mu\kappa}$ is a standard union bound while the last inequality is a standard bound on binomial coefficients.
Struct: (i) A 3-fold collision among the Q_{H_0} uniform A-tuples has probability at most $\binom{Q_{H_0}}{3}2^{-2(\kappa+\theta)}\le\frac{1}{6}Q_{H_0}^3 2^{-2(\kappa+\theta)}$; the same for the B-tuples, so the two overall probability is $\le\frac{1}{3}Q_{H_0}^3 2^{-2(\kappa+\theta)}$. (ii) Conditioned on all records other than x, the events "$A(x)$ equals another A-tuple" and "$B(x)$ equals another B-tuple" are independent (they depend on the fresh, mutually independent $A(x),B(x)$), each of probability $\le Q_{H_0}2^{-(\kappa+\theta)}$; a union bound over the Q_{H_0} choices of x gives $Q_{H_0}(Q_{H_0}2^{-(\kappa+\theta)})^2=Q_{H_0}^3 2^{-2(\kappa+\theta)}$. Summing, we finally get $\Pr[\mathsf{Struct}]\le\frac{4}{3}Q_{H_0}^3 2^{-2(\kappa+\theta)}$. □

Lemma 2 (Link structure). *Conditioned on* ¬Struct*: (a) distinct linked pairs use distinct (A-relevant, B-relevant) record pairs, and (b)* $L\le\frac{2}{3}Q_E$.

Proof. (a) If (x,ρ) and (x',ρ') shared both records then $A(x)=A(x')$, $B(x)=B(x')$ and $\rho=\rho'$, so x is doubly-collided with x'. (b) Form the bipartite simple graph with left nodes the selector-$(0,1)$ records, right nodes the selector-$(1,1)$ records, and one edge per linked pair (x,ρ) joining its A-relevant and B-relevant records. The degree of an A-relevant record $(z\|01,y\|\rho,c_0)$ is the number of x with $A(x)=(z,y)$ linked at the nonce ρ, hence at most the (z,y) multiplicity in $\{\!\{A(x),B(x)\,;\,(x,...)\in\mathcal{Q}_{H_0}\}\!\}$ which is ≤ 2 by ¬Struct(i); symmetrically for B-relevant records. On the other hand, if a node has degree 2, its two neighbours have degree 1 (otherwise some x would be doubly-collided). Hence every component with an edge is a star of $d\in\{1,2\}$ edges and $d{+}1$ nodes, giving a ratio at most $\frac{2}{3}$ for the number of edges (L) over the number of nodes (each cipher record is a distinct node making $\le Q_E$ nodes in total). We deduce $L\le\frac{2}{3}Q_E$. □

Lemma 3 (Certification rates). *We have:*

(a) For any forward cipher query $E(k,u)$ with answer v (i.e., leading to record (k,u,v)) and for every $v'\in\{0,1\}^\lambda$, we have:

$$\Pr[[v\oplus v']_{w-1}=0]\le 2^{-(w-1)}+t_k\,2^{-\lambda}$$

over the randomness of the answer v, where t_k is the number of images already assigned under k in the transcript prefix.

(b) For any backward cipher query $E^{-1}(k,v)$ with $k=z\|j\|1$ and the transcript prefix satisfying ¬MultKey*, the expected number of (x,ρ) pairs completed and certified by this query is at most $\frac{\mu\,2^\ell}{2^\lambda-Q_E}$.*

(c) For any H_0 query, the expected number of (x,ρ) pairs completed and certified by this query is at most $\frac{Q_E^2}{4}\cdot 2^{-2(\kappa+\theta)}$.

Proof. (a) We use the lazy-sampling view of the ideal cipher: the answer to the fresh forward query is drawn uniformly over the *full* $\{0,1\}^\lambda$, and resampled only on the *collision* event that it hits one of the t_k images already assigned under k (probability $t_k2^{-\lambda}$). For a full-domain uniform v, exactly $2^{\lambda-w+1}$ of the 2^λ values satisfy $[v\oplus v']_{w-1}=0$, so that probability is $2^{-(w-1)}$; bounding $\Pr[[v\oplus v']_{w-1}=0]\le\Pr[\text{fresh uniform in coset}]+\Pr[\mathsf{collision}]$ gives $2^{-(w-1)}+t_k2^{-\lambda}$.

(b) Say $j=0$; write the query as $E^{-1}(z\|01,v)$, answer $u=y\|\rho$ uniform over the $\ge 2^\lambda-Q_E$ unassigned preimages of $E_{z\|01}$. Because this is the *last* of the three records, a pair (x,ρ) completed by it has its H_0 record and its B-relevant partner already in the prefix, and satisfies $A(x)=(z,y)$. The set T of pairs it can complete-and-certify is thus

$$T=\{(x,\rho):(x,(g_0,g_1,k_0,k_1))\in\mathcal{Q}_{H_0},\ k_0=z,\ (k_1\|11,\ g_1\|\rho,\ c_1)\in\mathcal{Q}_E,\ [v\oplus c_1]_{w-1}=0\},$$

Each $(x, \rho) \in T$ requires the specific answer $u = g_0 \| \rho$ so the expected number completed is at most $|T|/(2^\lambda - Q_E)$. Finally, $|T| \le \mu\, 2^\ell$: the candidate x with $k_0 = z$ number at most μ by $\neg\mathsf{MultKey}$, and each contributes at most 2^ℓ nonces ρ.

(c) A pair (x, ρ) completed by the H_0 query on x has both cipher records in the prefix: a same-nonce, opposite-selector pair $\big((z\|01, y\|\rho, v),\ (z'\|11, y'\|\rho, v')\big)$ with $[v \oplus v']_{w-1} = 0$, such that the fresh answer gives $A(x) = (z, y)$ and $B(x) = (z', y')$. As $A(x), B(x)$ are uniform independent on $\{0,1\}^{\kappa+\theta}$, each candidate record pair is hit with probability $2^{-2(\kappa+\theta)}$; there are at most $Q_E^2/4$ same-nonce opposite-selector record pairs. □

Lemma 4 (Certified pairs). *Let $N = N_E + N_H$ where N_E counts certified pairs completed by a cipher record and N_H counts certified pairs completed by the H_0 record. Then*

$$\mathbb{E}[N_E \cdot \mathbf{1}_{\neg\mathsf{MultKey}\wedge\neg\mathsf{Struct}}] \;\le\; \underbrace{\frac{4}{3}\frac{Q_E}{2^w}}_{\text{fwd "ideal"}} \;+\; \underbrace{\frac{\mu\, 2^\ell\, Q_E}{2^\lambda}}_{\text{fwd "collision"}} \;+\; \underbrace{\frac{\mu\, 2^\ell\, Q_E}{2^\lambda - Q_E}}_{\text{backward}} ,$$

$$\mathbb{E}[N \cdot \mathbf{1}_{\neg\mathsf{MultKey}\wedge\mathsf{Struct}}] \;\le\; \frac{\mu Q_E}{2} \cdot \Pr[\mathsf{Struct}] \le \frac{\mu Q_E Q_{H_0}^3}{2^{2(\kappa+\theta)}} ,$$

$$\mathbb{E}[N_H] \;\le\; \frac{Q_{H_0} Q_E^2}{2^{2(\kappa+\theta)+2}} .$$

Proof. The lemma's principle is to charge each certified pair to its completing query.

First bound. For a certified pair completed by a cipher query, the H_0 record and the partner E record are already present and the linked pairs it may complete are determined by the transcript prefix. *Forward completions:* By Lemma 3(a) each completed link is certified with probability $\le 2^{-(w-1)} + t_k 2^{-\lambda}$. The $2^{-(w-1)}$ part, summed over linked pairs with the bound $L \cdot \mathbf{1}_{\neg\mathsf{Struct}} \le \frac{2}{3} Q_E$ from Lemma 2, gives $\frac{2}{3} Q_E \cdot 2^{-(w-1)} = \frac{4}{3} Q_E / 2^w$ (the "ideal" term). The "collision" part $\sum_{\text{linked pairs}} t_k 2^{-\lambda}$ can be grouped by the completing key k and using that each key hosts at most $\mu 2^\ell$ linked pairs (at most μ values of x matching k by $\neg\mathsf{MultKey}$, times 2^ℓ nonces), we get $\sum_{\text{linked pairs}} t_k 2^{-\lambda} \le \sum_k \mu 2^\ell Q_E^{(k)} 2^{-\lambda}$ where $Q_E^{(k)}$ denotes the number of E queries with key k. Since $\sum_k Q_E^{(k)} \le Q_E$, we obtain the upper bound $\mu 2^\ell Q_E 2^{-\lambda}$ for the "collision" term. *Backward completions:* Lemma 3(b) bounds each backward query's expected certifications by $\mu\, 2^\ell/(2^\lambda - Q_E)$; summing over $\le Q_E$ queries gives the third term.

Second bound. Under $\neg\mathsf{MultKey}$, each cipher record witnesses at most μ linked pairs and each linked pair is witnessed by two cipher records, so the number of linked pairs is at most $\mu Q_E/2$. Hence $N \le L \le \mu Q_E/2$, which multiplied by $\mathbf{1}_{\mathsf{Struct}}$ and taking the expectation gives the second bound.

Third bound. Sum Lemma 3(c) over the Q_{H_0} queries. □

Proof (Proof of Theorem 1). For the winning pair (x^*, ρ^*), let c_0^*, c_1^* be the two ciphertexts for which $\mathsf{Check}(x^*, \rho^*)$ succeeds, and let $s^* := x^* \| \rho^* \| c_0^* \| c_1^*$ be the corresponding input to H_1, so that the winning condition reads $H_1(s^*) \in \mathcal{H}_{x^*}$. Define the *pre-screening* event

$$\mathsf{Pre} \;:\; \text{the query } H_1(s^*) \text{ is made strictly before } (x^*, \rho^*) \text{ is certified,}$$

i.e. strictly before its completing record enters the transcript. We split the probability of the winning event Win according to whether Pre occurs or not:

$$\Pr[\mathsf{Win}] \;\le\; \Pr[\mathsf{Win} \wedge \neg\mathsf{Pre}] \;+\; \Pr[\mathsf{Win} \wedge \mathsf{Pre}] .$$

Pre-screened wins. On Pre, we charge the win to the query $H_1(s^*) \in \mathcal{Q}_{H_1}$. Fix any query $H_1(s) \in \mathcal{Q}_{H_1}$ with $s = x\|\rho\|c_0\|c_1$. For it to yield a pre-screened win, two events must occur:

(i) its answer lies in $\mathcal{H}_x$: as s is queried at most once, this holds with probability at most $|\mathcal{H}_x|/2^{2\lambda} \le \varepsilon$ over the random oracle H_1;

(ii) the pair (x, ρ) is subsequently certified with these very ciphertexts c_0, c_1: since the corresponding records are absent when $H_1(s)$ is made, its completing record is created *after* $H_1(s)$ and must realize the guessed configuration. If this record is a cipher record, its fresh answer must equal a prescribed value ($E(k_j\|\cdot,\, g_j\|\rho) = c_j$ or $E^{-1}(k_j\|\cdot,\, c_j) = g_j\|\rho$), of probability at most $1/(2^\lambda - Q_E)$; if it is

the H_0 record, its fresh answer must satisfy $A(x) = (k_0, g_0)$ and $B(x) = (k_1, g_1)$, of probability $2^{-2(\kappa+\theta)} \leq 1/(2^\lambda - Q_E)$. A union bound over the (at most five) possible completing records gives probability at most $5/(2^\lambda - Q_E)$.

Events (i) and (ii) depend on the independent oracles H_1 and (E, H_0), so each query to H_1 yields a pre-screened win with probability at most $5\varepsilon/(2^\lambda - Q_E)$. A union bound over the at most Q_{H_1} queries to H_1 gives

$$\Pr[\mathsf{Win} \wedge \mathsf{Pre}] \;\leq\; \varepsilon \, \frac{5\, Q_{H_1}}{2^\lambda - Q_E} = \varepsilon \, \frac{5\,(1+\alpha)\, Q_{H_1}}{2^\lambda}$$

(recalling $\alpha := Q_E/(2^\lambda - Q_E)$).

Non-pre-screened wins. On $\neg\mathsf{Pre}$, the query $H_1(s^*)$ is made after the certification of (x^*, ρ^*); its answer is then a value of H_1 drawn independently of the (already determined) certification event, and lies in $\mathcal{H}_{x^*}$ with probability at most ε. Charging each such win to the certified pair it uses and splitting by the type of completing record yields

$$\Pr[\mathsf{Win} \wedge \neg\mathsf{Pre}] \;\leq\; \Pr[\mathsf{Win}_E \cup \mathsf{Win}_H] \leq \Pr[\mathsf{Win}_E] + \Pr[\mathsf{Win}_H] \;,$$

where Win_E is the event that the completing record is a cipher record, and Win_H is the event that the completing record is the H_0 record, respectively satisfying $\Pr[\mathsf{Win}_E] \leq \varepsilon\, \mathbb{E}[N_E]$ and $\Pr[\mathsf{Win}_H] \leq \varepsilon\, \mathbb{E}[N_H]$. We have:

- Win_E happens with probability ε for each certified pair completed by a cipher record:

$$\begin{aligned}\Pr[\mathsf{Win}_E] \leq \varepsilon\, \mathbb{E}[N_E] &\leq \varepsilon\, \mathbb{E}[N_E \cdot \mathbf{1}_{\neg\mathsf{MultKey}\wedge\neg\mathsf{Struct}}] + \varepsilon\, \mathbb{E}[N \cdot \mathbf{1}_{\neg\mathsf{MultKey}\wedge\mathsf{Struct}}] + \Pr[\mathsf{MultKey}] \\ &\leq \varepsilon \left(\frac{4}{3}\frac{Q_E}{2^w} + \frac{\mu 2^\ell Q_E}{2^\lambda} + \frac{\mu 2^\ell Q_E}{2^\lambda - Q_E} + \frac{\mu Q_E Q_{H_0}^3}{2^{2(\kappa+\theta)}} \right) + p_{\mu+1}.\end{aligned}$$

  The above inequalities directly follow from Lemmas 1 and 4.
- Win_H happens with probability ε for each certified pair completed by its H_0 record:

$$\Pr[\mathsf{Win}_H] \leq \varepsilon\, \mathbb{E}[N_H] \leq \varepsilon\, \frac{Q_{H_0} Q_E^2}{2^{2(\kappa+\theta)+2}} \;.$$

Adding the three upper bounds concludes the proof. □

5 Variants

5.1 Generalization to $t = 2^\tau$ Cipher Calls per Iteration

The scheme extends to any number $t = 2^\tau$ of cipher calls per iteration. One derives t tuples

$$\big((k_0, g_0), \ldots, (k_{t-1}, g_{t-1})\big) = H_0(x) \;\in \{0,1\}^{t(\kappa+\theta)}$$

computes $c_j = E\big(k_j \,\|\, \langle j \rangle \,\|\, 1,\; g_j \| \rho\big)$ for $j = 0, \ldots, t-1$ (where $\langle j \rangle$ is the τ-bit index of the call and the last bit is the selector shared with the signature scheme), and accepts when

$$\Big[\, \textstyle\bigoplus_{j=0}^{t-1} c_j \,\Big]_{w-\tau} = 0 \;.$$

The condition now bears on $w - \tau$ bits, so an iteration succeeds with probability $2^{-(w-\tau)}$ and costs t calls; the expected grinding cost is $t \cdot 2^{w-\tau} = 2^w$ cipher calls, *independently of* t. Likewise, correctness holds as soon as $\ell \geq w - \tau + \log_2(\lambda \ln 2)$.

Two things change in the analysis. First, indexing the t calls requires $\tau+1$ selector bits, so $\kappa = \lambda - \tau - 1$ and the tuple binding shrinks to $\kappa + \theta = 2\lambda - \ell - \tau - 1$. Second, a linked pair is now witnessed by t cipher records (one per call), and the link-structure lemma generalizes: under $\neg\mathsf{Struct}$, the link hypergraph decomposes into isolated hyperedges (t records for 1 link) and 2-hyperedge stars sharing a single record ($2t-1$ records for 2 links), whence $L \leq \frac{2}{2t-1}\, Q_E$. Charging the certification of a forward-completed pair

to its fresh last record — whose answer must hit a coset of relative size $2^{-(w-\tau)}$ — the leading term becomes

$$\frac{2}{2t-1}\,Q_E \cdot 2^{-(w-\tau)} \;=\; \frac{2t}{2t-1}\cdot\frac{Q_E}{2^w}\;.$$

Thus Theorem 1 holds with leading term $\varepsilon \cdot \frac{2t}{2t-1} \cdot \frac{Q_E}{2^w}$; the residual terms are unchanged up to the substitution $\kappa = \lambda - \tau - 1$ (which marginally enlarges the still-negligible structural terms) and the replacement of the constant 5 in the pre-screening term by $2t+1$ (the number of possible completing queries).

The leading constant $\frac{2t}{2t-1} = \frac{2^{\tau+1}}{2^{\tau+1}-1}$ decreases from 2 ($t=1$) to $\frac{4}{3}$ ($t=2$), $\frac{8}{7}$ ($t=4$), $\frac{16}{15}$ ($t=8$), ..., tending to 1 as t grows: more calls per iteration bring the adversary's advantage closer to the ideal $Q_E\varepsilon/2^w$, at the cost of τ extra selector bits (a smaller κ), a longer H_0 output ($t(\kappa+\theta)$ bits) and more cipher calls per iteration. The case $t=2$ already captures most of the gain (a 0.42-bit loss instead of 1), which is why we focus on it.

5.2 Configuration 2: Selector Bit in the Plaintext

We now consider the second configuration listed in Section 3, where the block cipher calls of the signature scheme are made with a *secret* key k and a plaintext $u = \mathsf{salt}\|\mathsf{tweak}$. Since the domain-separation selector belongs to the public tweak, it must now be placed in the *plaintext* argument: calls of the signature scheme use plaintexts ending with the bit 0, and calls of the grinding scheme plaintexts ending with the bit 1. The signature scheme therefore sacrifices one bit of its tweak (rather than one bit of salt in the key).

The scheme. For a string x, parse

$$H_0(x) = (g_0, g_1, k_0, k_1) \in \{0,1\}^\theta \times \{0,1\}^\theta \times \{0,1\}^\kappa \times \{0,1\}^\kappa, \qquad \kappa = \lambda - 1, \quad \theta = \lambda - \ell - 1,$$

and define, for a nonce $\rho \in \{0, \ldots, 2^\ell - 1\}$, the two calls

$$c_0 := E\big(k_0\|0,\ g_0\|\rho\|1\big), \qquad c_1 := E\big(k_1\|1,\ g_1\|\rho\|1\big).$$

The success condition $[c_0 \oplus c_1]_{w-1} = 0$ and the digest $h := H_1(x\|\rho\|c_0\|c_1)$ are unchanged. The scheme is otherwise identical to that of Section 3. Note that $\kappa + \theta = 2\lambda - \ell - 2$, the same value as for the scheme of Section 3: the size of the tuples binding a pair (x, ρ) depends only on the number of bits spent on the nonce and the selectors, not on the argument in which they are spent. The scheme is depicted in Figure 2.

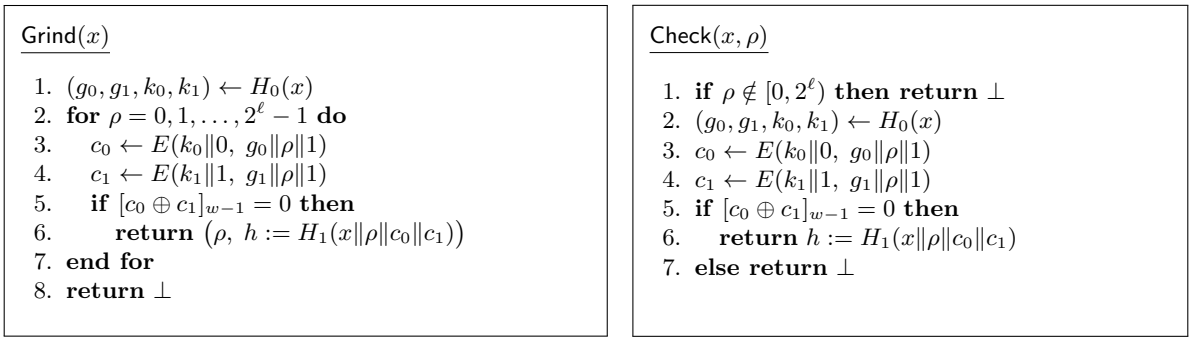

Fig. 2. Grinding scheme for configuration 2 (selector bit in the plaintext).

Changes in the proof. The proof of Theorem 1 applies with the following modifications only.

– *Relevance.* In the paragraph "Tuples, links, certification", a cipher record (k, u, v) with $k = z\|0$ and $u = y\|\rho\|1$ (key part $z \in \{0,1\}^\kappa$, block part $y \in \{0,1\}^\theta$, nonce ρ) is *A-relevant for x at ρ* if $(z, y) = A(x)$, and B-relevance is defined symmetrically with $k = z\|1$. The rest of the paragraph is unchanged; in particular the A- and B-relevant records of a linked pair are still distinguished by their *key*, so the bipartite graph of Lemma 2 is built exactly as before (left nodes: records with key ending in 0; right nodes: key ending in 1) and the bound $L \le \frac{2}{3}Q_E$ is unaffected. Likewise, the two calls of an iteration still use distinct keys, so Remark 1 applies verbatim.
– *Conventions.* The second item of the conventions is replaced by: *cipher queries whose plaintext ends with the bit 0 are never A- or B-relevant, hence never witness a linked pair; unlike in the main scheme they cannot be discarded, as they are addressed to the same permutations as the grinding queries and consume part of their domain and range. They are therefore counted in Q_E, and the quantity t_k of Lemma 3(a) counts the images assigned under k by queries of either kind.*
– *Bad events and certification rates.* MultKey now bears on $\kappa = \lambda - 1$ bits, and in Lemma 3 the target answer of a backward query reads $u = g_0\|\rho\|1$ while the candidate record pairs of item (c) are the same-nonce pairs $\big((z\|0,\, y\|\rho\|1,\, v),\, (z'\|1,\, y'\|\rho\|1,\, v')\big)$ with $[v \oplus v']_{w-1} = 0$. The statements and bounds are unchanged, and so are Lemma 4 and the pre-screening analysis.
– *Accounting of the foreign queries.* The sharing of the permutations between the grinding scheme and the signature scheme does not affect any bound. Indeed, the sole effect of a selector-0 query is to consume one input-output pair of the permutation it addresses, which is precisely what the terms $t_k 2^{-\lambda}$ and the denominators $2^\lambda - Q_E$ already account for: Lemma 3(a) holds verbatim with the larger t_k, the "collision" term of Lemma 4 is still bounded by $\sum_k \mu 2^\ell Q_E^{(k)} 2^{-\lambda} \le \mu 2^\ell Q_E 2^{-\lambda}$ since $\sum_k Q_E^{(k)} \le Q_E$ now counts all cipher queries, and the $\ge 2^\lambda - Q_E$ unassigned preimages of Lemma 3(b) are counted with respect to the same total budget.

As $\kappa + \theta$ is unchanged, the bound of Theorem 1 holds verbatim, with $\kappa = \lambda - 1$ and $\theta = \lambda - \ell - 1$.

Independence of the bad-challenge family. A last point concerns the security model rather than the proof. In Definition 1 the family $\{\mathcal{H}_x\}_x$ is fixed independently of the oracles. In the composed setting, $\mathcal{H}_x$ is determined by the commitment contained in x, which might be computed from selector-0 cipher calls; with a selector bit in the key these concerned independent permutations, whereas they now concern the same ones. The definition must therefore be strengthened by letting $\{\mathcal{H}_x\}_x$ be an arbitrary function of the selector-0 *records* of the transcript (the query-response pairs with plaintext ending in 0), subject to $|\mathcal{H}_x| \le \varepsilon \cdot 2^{2\lambda}$. The proof then conditions on these records: given any such conditioning, the answers to the grinding queries remain uniform over the values not yet assigned under their key, which is precisely the lazy-sampling view used in Lemma 3, the induced bias being absorbed by the terms $t_k 2^{-\lambda}$. The dependence must be restricted to the *queried* points: allowing $\{\mathcal{H}_x\}_x$ to depend on the whole restriction of E to selector-0 plaintexts would fix half of the range of every permutation, and the resulting bias on the grinding answers could no longer be bounded by $t_k 2^{-\lambda}$.

Remark 3 (Combination with the t-call variant). The two variants combine: one uses $t = 2^\tau$ calls $c_j := E\big(k_j\|\langle j\rangle,\ g_j\|\rho\|1\big)$ for $j = 0, \ldots, t-1$, with the τ-bit call index in the key and the separation bit in the plaintext, and the success condition $\big[\bigoplus_j c_j\big]_{w-\tau} = 0$. The parameters become $\kappa = \lambda - \tau$ and $\theta = \lambda - \ell - 1$, so that

$$\kappa + \theta \;=\; 2\lambda - \ell - (\tau + 1),$$

which is the same value as in Configuration 1 (which only depends on the ℓ nonce bits and the $\tau + 1$ selector bits). The bound is then the one of Section 5.1, with leading term $\varepsilon \cdot \frac{2t}{2t-1} \cdot \frac{Q_E}{2^w}$, together with the two adaptations described above (foreign queries counted in Q_E and t_k, and dependence of $\{\mathcal{H}_x\}_x$ on the selector-0 records).

Acknowledgments

The author thanks Ryad Benadjila, Charles Bouillaguet and Thibauld Feneuil for insightful discussions on the topic of this work.

References

ABB+24. Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Nicolas Gama, Shay Gueron, James Howe, Andreas Hülsing, David Joseph, Antoine Joux, Mukul Kulkarni, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH — Syndrome Decoding in the Head. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

BBB+24. Carsten Baum, Lennart Braun, Ward Beullens, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

BBD+23. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 581–615. Springer, Cham, August 2023.

BBFR24. Ryad Benadjila, Charles Bouillaguet, Thibauld Feneuil, and Matthieu Rivain. MQOM — MQ on my Mind. Technical report, National Institute of Standards and Technology, 2024. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures`.

BBM+24. Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. One tree to rule them all: Optimizing GGM trees and OWFs for post-quantum signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 463–493. Springer, Singapore, December 2024.

FR25. Thibauld Feneuil and Matthieu Rivain. Threshold computation in the head: Improved framework for post-quantum signatures and zero-knowledge arguments. *Journal of Cryptology*, 38(3):28, July 2025.

Sta21. StarkWare. ethSTARK documentation. Cryptology ePrint Archive, Report 2021/582, 2021.